# Restful-booker API Kapsamlı Test Raporu

## 1. Giriş

### 1.1. Raporun Amacı

Bu rapor, Restful-booker API'si için gerçekleştirilen kapsamlı test çalışmalarının sonuçlarını sunmak amacıyla hazırlanmıştır. Rapor, API'nin detaylı bir analizini, uygulanan test senaryolarını, bu senaryolara ait test sonuçlarını ve API ile etkileşimde bulunan bir kullanıcı arayüzü (UI) için test stratejisini içermektedir.

### 1.2. Test Edilen API Hakkında Genel Bilgi

Test edilen API, **Restful-booker** . Bu API, temel olarak Oluşturma (Create), Okuma (Read), Güncelleme (Update) ve Silme (Delete) (CRUD) işlemlerini destekleyen bir Web API'sidir ve kimlik doğrulama özelliklerine sahiptir. Özellikle API testi öğrenimi ve çeşitli test araçlarının denenmesi amacıyla geliştirilmiş olup, keşfedilmesi için bir dizi hata içermektedir.

API'nin canlı (deployed) bir örneğine [https://restful-booker.herokuapp.com/](https://restful-booker.herokuapp.com/%20) adresinden erişilebilmektedir. API dokümantasyonuna ise [https://restful-booker.herokuapp.com/apidoc/index.html](https://restful-booker.herokuapp.com/apidoc/index.html%20) adresinden ulaşılabilir.

API'nin önemli bir özelliği, her 10 dakikada bir varsayılan durumuna (önceden yüklenmiş 10 kayıt ile) sıfırlanmasıdır.

### 1.3. Test Kapsamı

Bu testteki temel endpoint'ler:

* **Auth:** Kimlik doğrulama ve token üretme işlemleri.
* **Booking:** Rezervasyon oluşturma, getirme, güncelleme, kısmi güncelleme ve silme işlemleri.
* **Ping:** API'nin sağlık durumunu kontrol etme işlemleri.

Testler, fonksiyonel testleri (CRUD operasyonları), kimlik doğrulama testlerini ve temel sağlık kontrolü (Ping) testlerini kapsamaktadır

### 1.4. Test Ortamı ve Araçları

* **Test Ortamı:** Testler, Restful-booker API'sinin [https://restful-booker.herokuapp.com/](https://restful-booker.herokuapp.com/%20) adresinde bulunan halka açık, canlı örneği üzerinde gerçekleştirilmiştir.
* **Ana Test Aracı:** API isteklerinin gönderilmesi, yanıtların doğrulanması ve testlerin otomatize edilmesi için **Python Requests Kütüphanesi** kullanılmıştır.

## 2. API Analizi

### 2.1. API Endpoint'leri ve Metotları

Restful-booker API'sinin temel endpoint'leri ve desteklediği HTTP metotları aşağıda özetlenmiştir.

|  |  |  |
| --- | --- | --- |
| **Endpoint Yolu** | **HTTP Metodu** | **Açıklama** |
| /auth | POST | Kimlik doğrulama için token oluşturur. |
| /booking | GET | Tüm rezervasyon ID'lerini veya filtrelenmiş ID'leri getirir (isme veya tarihe göre). |
| /booking/{id} | GET | Belirli bir rezervasyonun detaylarını getirir. |
| /booking | POST | Yeni bir rezervasyon oluşturur (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | PUT | Mevcut bir rezervasyonu tamamen günceller (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | PATCH | Mevcut bir rezervasyonu kısmen günceller (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | DELETE | Belirli bir rezervasyonu siler. |
| /ping | GET | API'nin sağlık durumunu kontrol eder. |

### 2.2. İstek ve Yanıt Formatları

Restful-booker API'si, istek ve yanıt gövdeleri için öncelikli olarak **JSON** formatını kullanmaktadır.

* İsteklerde genellikle Content-Type: application/json başlığı kullanılır.
* Bazı isteklerde, özellikle PUT gibi, yanıt olarak JSON beklendiğini belirtmek için Accept: application/json başlığı da kullanılmaktadır.

### 2.3. Kimlik Doğrulama Mekanizması

API, belirli endpoint'ler (özellikle *PUT /booking/{id}*, *PATCH /booking/{id}* ve *DELETE /booking/{id})* için kimlik doğrulama gerektirir. İki tür kimlik doğrulama yöntemi desteklenmektedir :

1. **Token Tabanlı Kimlik Doğrulama:**
   * POST /auth endpoint'ine username ("admin") ve password ("password123") bilgileri gönderilerek bir kimlik doğrulama token'ı alınır.
   * Alınan bu token, korumalı endpoint'lere yapılan isteklerde Cookie başlığı içinde (token=ALINAN\_TOKEN\_DEGERI formatında) gönderilir.
2. **Basic Auth (Temel Kimlik Doğrulama):**
   * Korumalı endpoint'lere yapılan isteklerde, Authorization başlığı altında Basic Auth yöntemiyle admin kullanıcı adı ve password123 şifresi kullanılarak kimlik doğrulaması yapılabilir.

### 2.4. Durum Kodları ve Hata Yönetimi

API'nin çeşitli durumlar için döndürmesi beklenen HTTP durum kodları şunlardır:

* **Başarılı İşlemler:**
  + 200 OK: Başarılı GET, PUT, PATCH istekleri için.
  + 201 Created: Başarılı POST (oluşturma) ve bazı dokümanlara göre DELETE ve Ping istekleri için (Bu durumlar test edilerek doğrulanmalıdır, zira DELETE için 204 No Content veya 200 OK, Ping için 200 OK daha yaygındır).
* **İstemci Hataları:**
  + 400 Bad Request: Geçersiz istek formatı veya eksik parametreler gibi durumlarda.
  + 401 Unauthorized: Geçerli kimlik bilgileri olmadan korumalı bir kaynağa erişmeye çalışıldığında (genellikle token eksik veya geçersizse).
  + 403 Forbidden: Kimlik doğrulanmış olsa bile, kullanıcının belirli bir kaynağa erişim yetkisi olmadığında.3
  + 404 Not Found: İstenen kaynak bulunamadığında (örneğin, var olmayan bir rezervasyon ID'si ile GET isteği).

## 3. Test Senaryoları ve Uygulanışı

Aşağıda, API'nin temel işlevlerini doğrulamak amacıyla tasarlanan ve uygulanan test senaryoları detaylandırılmıştır.

### 3.1. Auth Endpoint Testleri

#### TS001: Başarılı Token Oluşturma

* **Açıklama:** Geçerli kullanıcı adı ve şifre ile /auth endpoint'ine POST isteği göndererek başarılı bir şekilde token alınmasını test eder.
* **Adımlar:**
  1. POST isteğini /auth endpoint'ine gönder.
  2. İstek gövdesine aşağıdaki JSON:

JSON

{

    "username" : "admin",

    "password" : "password123"

}

* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Bir token alanı içeren JSON nesnesi.

#### TS002: Başarısız Token Oluşturma - Yanlış Kimlik Bilgileri

* **Açıklama:** Geçersiz kullanıcı adı veya şifre ile /auth endpoint'ine POST isteği gönderildiğinde token oluşturulamamasını test eder.
* **Adımlar:**
  1. POST isteğini /auth endpoint'ine gönder.
  2. İstek gövdesine aşağıdaki gibi geçersiz bilgiler içeren JSON:
* JSON
* {
* "username" : "invalid\_user",
* "password" : "invalid\_password"
* }

* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK (API'nin hatalı davranışına göre) veya daha standart bir hata kodu (400 Bad Request / 401 Unauthorized).
  + Yanıt Gövdesi: Hata mesajı içeren bir JSON nesnesi

### 3.2. Booking Endpoint Testleri

#### CRUD Operasyonları

**TS003: Yeni Rezervasyon Oluşturma (POST /booking)**

* **Açıklama:** Geçerli verilerle /booking endpoint'ine POST isteği göndererek yeni bir rezervasyon oluşturulmasını test eder.
* **Adımlar:**
  1. POST isteğini /booking endpoint'ine gönder.
  2. Content-Type: application/json başlığını ekle.
  3. İstek gövdesine 3'deki gibi geçerli bir rezervasyon JSON:
* JSON
* {
* "firstname" : "Jim",
* "lastname" : "Brown",
* "totalprice" : 111,
* "depositpaid" : true,
* "bookingdates" : {
* "checkin" : "2023-01-01",
* "checkout" : "2023-01-10"
* },
* "additionalneeds" : "Breakfast"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: **201 Created.**
  + Yanıt Gövdesi: Oluşturulan rezervasyonun detaylarını ve bir bookingid içeren JSON nesnesi.

**TS004: Rezervasyon Bilgilerini Getirme (GET /booking/{id})**

* **Açıklama:** TS003'te oluşturulan bookingid kullanılarak rezervasyon detaylarının getirilmesini test eder.
* **Adımlar:**
  1. GET isteğini /booking/{{bookingId}} endpoint'ine gönder ({{bookingId}}).
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: İlgili bookingid'ye ait rezervasyon detaylarını içeren JSON nesnesi.

**TS005: Rezervasyonu Güncelleme (PUT /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun bilgilerinin PUT metodu ile güncellenmesini test eder. Bu işlem kimlik doğrulama gerektirir.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak *Cookie: token={{authToken}}* başlığı.
  2. *Content-Type: application/json* ve *Accept: application/json* başlıkları.
  3. PUT isteğini /booking/{{bookingId}} endpoint'ine.
  4. İstek gövdesine güncellenmiş rezervasyon bilgilerini içeren JSON:
* JSON
* {
* "firstname" : "James",
* "lastname" : "Brown",
* "totalprice" : 111,
* "depositpaid" : true,
* "bookingdates" : {
* "checkin" : "2023-01-01",
* "checkout" : "2023-01-15"
* },
* "additionalneeds" : "Lunch"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Güncellenmiş rezervasyon detaylarını içeren JSON nesnesi.

**TS006: Rezervasyonu Kısmi Güncelleme (PATCH /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun sadece belirli bilgilerinin PATCH metodu ile güncellenmesini test eder. Bu işlem kimlik doğrulama gerektirir.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak Cookie: token={{authToken}} başlığı.
  2. Content-Type: application/json ve Accept: application/json başlıkları.
  3. PATCH isteğini /booking/{{bookingId}} endpoint'e.
  4. İstek gövdesine sadece güncellenecek alanları içeren JSON:
* JSON
* {
* "firstname" : "Jimmy",
* "additionalneeds" : "Dinner"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Kısmen güncellenmiş rezervasyon detaylarını içeren JSON nesnesi.

**TS007: Rezervasyonu Silme (DELETE /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun DELETE metodu ile silinmesini test eder.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak Cookie: token={{authToken}} başlığı ile.
  2. DELETE isteğini /booking/{{bookingId}} endpoint'ine.
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 201 Created

**TS008: Silinen Rezervasyonu Getirme (Attempt to Get Deleted Booking)**

* **Açıklama:** TS007'de silinen rezervasyonun ID'si ile GET isteği yapıldığında bulunamaması gerektiğini test eder.
* **Adımlar:**
  1. GET isteğini /booking/{{bookingId}} endpoint'ine (silinen rezervasyonun ID'si ile).
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 404 Not Found.

#### Filtreleme ve Diğer GET Testleri

**TS009: Tüm Rezervasyon ID'lerini Getirme (GET /booking)**

* **Açıklama:** /booking endpoint'ine GET isteği göndererek tüm rezervasyon ID'lerinin listesini almayı test eder.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi bookingid içeren bir dizi olmalıdır.

**TS010: Rezervasyonları İsimle Filtreleme (GET /booking?firstname=X&lastname=Y)**

* **Açıklama:** firstname ve lastname query parametreleri kullanılarak rezervasyonların filtrelenmesini test eder.
* **Adımlar:**
  1. Önceden bilinen veya oluşturulmuş bir rezervasyonun firstname ve lastname bilgilerini kullanarak GET /booking?firstname=TestAd&lastname=TestSoyad isteği.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi sadece eşleşen rezervasyonların ID'lerini içermelidir.

**TS011: Rezervasyonları Tarihle Filtreleme (GET /booking?checkin=X&checkout=Y)**

* **Açıklama:** checkin ve checkout query parametreleri kullanılarak rezervasyonların tarih aralığına göre filtrelenmesini test eder.
* **Adımlar:**
  1. Belirli bir tarih aralığı (YYYY-MM-DD formatında) belirleyerek GET /booking?checkin=2023-01-01&checkout=2023-01-31 isteği.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi belirtilen tarih aralığındaki rezervasyonların ID'lerini içermelidir.

#### Negatif Senaryolar ve Hata Durumları

**TS012: Geçersiz ID ile Rezervasyon Getirme (Get Booking with Invalid ID)**

* **Açıklama:** Var olmayan veya geçersiz formatta bir ID ile rezervasyon getirme girişiminin nasıl sonuçlandığını test eder.
* **Adımlar:**
  1. GET /booking/gecersiz999 isteği.
* **Beklenen Sonuç:** 404 Not Found.

**TS013: Eksik Alanlarla Rezervasyon Oluşturma (Create Booking with Missing Required Fields)**

* **Açıklama:** Yeni bir rezervasyon oluştururken zorunlu alanların eksik bırakılması durumunda API'nin nasıl tepki verdiğini test eder.
* **Adımlar:**
  1. POST /booking isteği ile zorunlu bir alanı eksik olan bir JSON gövdesi gönder.
* **Beklenen Sonuç:** 400 Bad Request.

**TS014: Yetkisiz Güncelleme/Silme Denemesi (Attempt Unauthorized Update/Delete)**

* **Açıklama:** Geçerli bir token olmadan veya Basic Auth bilgileri olmadan korumalı PUT, PATCH veya DELETE endpoint'lerine erişim denemelerini test eder.
* **Adımlar:**
  1. Cookie başlığı olmadan PUT /booking/{{bookingId}} isteği.
* **Beklenen Sonuç:** 403 Forbidden veya 401 Unauthorized.

### 3.3. Ping Endpoint Testleri

**TS015: API Sağlık Kontrolü (GET /ping)**

* **Açıklama:** /ping endpoint'ine GET isteği göndererek API'nin çalışır durumda ve erişilebilir olduğunu teyit eder.
* **Adımlar:**
  1. GET isteğini /ping endpoint'ine gönder.
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 201 Created (API dokümantasyonunda bu şekilde belirtilmişse; 200 OK daha yaygın bir durum kodudur ve doğrulanmalıdır).
  + Yanıt Gövdesi: Genellikle boş veya basit bir onay mesajı (örneğin, "Created" veya "OK").

### 3.4. Kod Kesitleri

* **Durum Kodunu Kontrol Etme:**
* JavaScript
* pm.test("Status code is 200", function () {
* pm.response.to.have.status(200);
* });

* **Yanıt Süresini Kontrol Etme (Basit Performans Kontrolü):**
* JavaScript
* pm.test("Response time is less than 500ms", function () {
* pm.expect(pm.response.responseTime).to.be.below(500);
* });

* **JSON Yanıtında Belirli Bir Alanın Varlığını ve Değerini Kontrol Etme:**
* JavaScript
* pm.test("Booking details are correct", function () {
* var jsonData = pm.response.json();
* pm.expect(jsonData.firstname).to.eql("James");
* pm.expect(jsonData.bookingdates.checkin).to.eql("2023-01-01");
* });
* **Yanıt Başlığını Kontrol Etme:**
* JavaScript
* pm.test("Content-Type header is present and correct", function () {
* pm.response.to.have.header("Content-Type");
* pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
* });

Bu script'ler, API yanıtlarının beklenen formatta ve içerikte olup olmadığını otomatik olarak doğrulamak için her bir test isteğinin "Tests" sekmesine eklenmiştir.

## 4. Test Sonuçları ve Analizi

### 4.1. Yürütülen Testlerin Özeti

Aşağıdaki tablo, yürütülen test senaryolarının bir özetini ve genel durumlarını göstermektedir. Detaylı analizler sonraki bölümlerde sunulmuştur.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Senaryosu ID** | **Açıklama** | **Durum (Başarılı/Başarısız/Kısmen Başarılı)** | **Notlar** |
| TS001 | Başarılı Token Oluşturma | Başarılı | Token alındı. |
| TS002 | Başarısız Token Oluşturma - Yanlış Kimlik Bilgileri | Başarılı | Beklendiği gibi "Bad credentials" mesajı ve 200 OK ile yanıt alındı (API'nin hatalı davranışı). |
| TS003 | Yeni Rezervasyon Oluşturma | Başarılı | Rezervasyon oluşturuldu, bookingid alındı. Yanıt kodu 200 OK (dokümantasyonda 201 bekleniyordu). |
| TS004 | Rezervasyon Bilgilerini Getirme | Başarılı | Oluşturulan rezervasyon detayları doğru getirildi. |
| TS005 | Rezervasyonu Güncelleme | Başarılı | Rezervasyon başarıyla güncellendi. |
| TS006 | Rezervasyonu Kısmi Güncelleme | Başarılı | Rezervasyon başarıyla kısmen güncellendi. |
| TS007 | Rezervasyonu Silme | Başarılı | Rezervasyon silindi. Yanıt kodu 201 Created (beklendiği gibi). |
| TS008 | Silinen Rezervasyonu Getirme | Başarılı | 404 Not Found yanıtı alındı. |
| TS009 | Tüm Rezervasyon ID'lerini Getirme | Başarılı | Rezervasyon ID listesi alındı. |
| TS010 | Rezervasyonları İsimle Filtreleme | Kısmen Başarılı | Filtreleme çalışıyor ancak boş olmayan lastname ile filtreleme yapıldığında beklenmedik sonuçlar. |
| TS011 | Rezervasyonları Tarihle Filtreleme | Başarılı | Tarih filtresi doğru çalışıyor. |
| TS012 | Geçersiz ID ile Rezervasyon Getirme | Başarılı | 404 Not Found yanıtı alındı. |
| TS013 | Eksik Alanlarla Rezervasyon Oluşturma | Başarılı | 400 Bad Request yerine 500 Internal Server Error alındı (Bu bir hata olarak değerlendirildi). |
| TS014 | Yetkisiz Güncelleme/Silme Denemesi | Başarılı | 403 Forbidden yanıtı alındı. |
| TS015 | API Sağlık Kontrolü (Ping) | Başarılı | 201 Created yanıtı alındı. |

## 6. Sonuç

### 6.1. Genel Test Değerlendirmesi

Restful-booker API'si üzerinde yürütülen testler, API'nin temel işlevlerinin büyük ölçüde çalıştığını göstermekle birlikte, bazı alanlarda iyileştirmelere ve dikkatli kullanıma ihtiyaç duyulduğunu ortaya koymuştur. API'nin birincil amacının API testi öğrenimi ve hataların keşfi için bir platform sağlamak olduğu göz önüne alındığında, karşılaşılan bazı "hatalar" aslında API'nin bu amacına hizmet eden kasıtlı özelliklerdir.